AKTU

B.Tech 6th Sem

GATEWAY CLASSES

CS IT & CS Allied

Software Engineering

Unit-5: One Shot

Pushpendra Sir

Software Engineering (BCS-601)

AKTU- Syllabus

Unit - IV

Software Maintenance and Software Project Management: Software as an Evolutionary Entity, Need for Maintenance, Categories of Maintenance: Preventive, Corrective and Perfective Maintenance, Cost of Maintenance, Software Re- Engineering, Reverse Engineering. Software Configuration Management Activities, Change Control Process, Software Version Control, An Overview of CASE Tools. Estimation of Various Parameters such as Cost, Efforts, Schedule/Duration, Constructive Cost Models (COCOMO), Resource Allocation Models, Software Risk Analysis and Management.

Software as an Evolutionary Entity

Software evolution is a concept that refers to the process of creating software and then upgrading it on a regular basis for different purpose such as adding new features or removing absolute functionality, change analysis, release preparation system implementation and system release to customer are all part of evolution process. The cost and the effect of this changes was examined to determine how much of the system will impacted by the reform and how much it will cost to enforce it. A new version of the software system is expected if the purpose improvements are approved all potential changes article into account during release preparation.

Importance of software evolution

- Change in requirement with time the organizations need and modulus operator of working changes significant overtime so that tools software they use must change to maximize the output in this rapidly changing time.

- Environment change when the working environment changes the item that allows us to operate in that environment change proportionally.

- Error and bugs as the age of organizations deployed software increase its precision decreases and its ability to handle increasingly Complex workload continue to degrade.

Software Maintenance

It is widely accepted part of SDLC. It stands for all the modifications and updation done after the delivery of software product. There are number of reasons, why modifications are required, some of them are briefly mentioned below:

- **Market Conditions** - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification. Client Requirements - Over the time, customer may ask for new features or functions in the software.
- **Host Modifications** - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.
- **Organization Changes** - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.

Types of Maintenance

In a software lifetime, type of maintenance may vary based on its nature. It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature. Following are some types of maintenance based on their characteristics:

Corrective Maintenance - This includes modifications and updations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.

Adaptive Maintenance - This includes modifications and updations applied to keep the software product up-to date and tuned to the ever changing world of technology and business environment.

Types of Maintenance.....

Perfective Maintenance - This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.

Preventive Maintenance - This includes modifications and updations to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future.

Cost of Maintenance

Reports suggest that the cost of maintenance is high. A study on estimating software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle. On an average, the cost of software maintenance is more than 50% of all SDLC phases. There are various factors, which trigger maintenance cost go high, such as:

Real-world factors affecting Maintenance Cost:

The standard age of any software is considered up to 10 to 15 years. Older softwares, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced softwares on modern hardware.

As technology advances, it becomes costly to maintain old software. Most maintenance engineers are newbie and use trial and error method to rectify problem. Often, changes made can easily hurt the original structure of the software, making it hard for any subsequent changes. Changes are often left undocumented which may cause more conflicts in future.

Cost of Maintenance.....

Software-end factors affecting Maintenance Cost

- Structure of Software Program
- Programming Language
- Dependence on external environment
- Staff reliability and availability

Maintenance Activities

IEEE provides a framework for sequential maintenance process activities. It can be used in iterative manner and can be extended so that customized items and processes can be included.

These activities go hand-in-hand with each of the following phase:

Identification & Tracing - It involves activities pertaining to identification of requirement of modification or maintenance. It is generated by user or system may itself report via logs or error messages. Here, the maintenance type is classified also.

Analysis - The modification is analyzed for its impact on the system including safety and security implications. If probable impact is severe, alternative solution is looked for. A set of required modifications is then materialized into requirement specifications. The cost of modification/maintenance is analyzed and estimation is concluded.

Design - New modules, which need to be replaced or modified, are designed against requirement specifications set in the previous stage. Test cases are created for validation and verification. Implementation - The new modules are coded with the help of structured design created in the design step.Every programmer is expected to do unit testing in parallel.

Maintenance Activities...

System Testing - Integration testing is done among newly created modules. Integration testing is also carried out between new modules and the system. Finally the system is tested as a whole, following regressive testing procedures.

Acceptance Testing - After testing the system internally, it is tested for acceptance with the help of users. If at this state, user complaints some issues they are addressed or noted to address in next iteration.

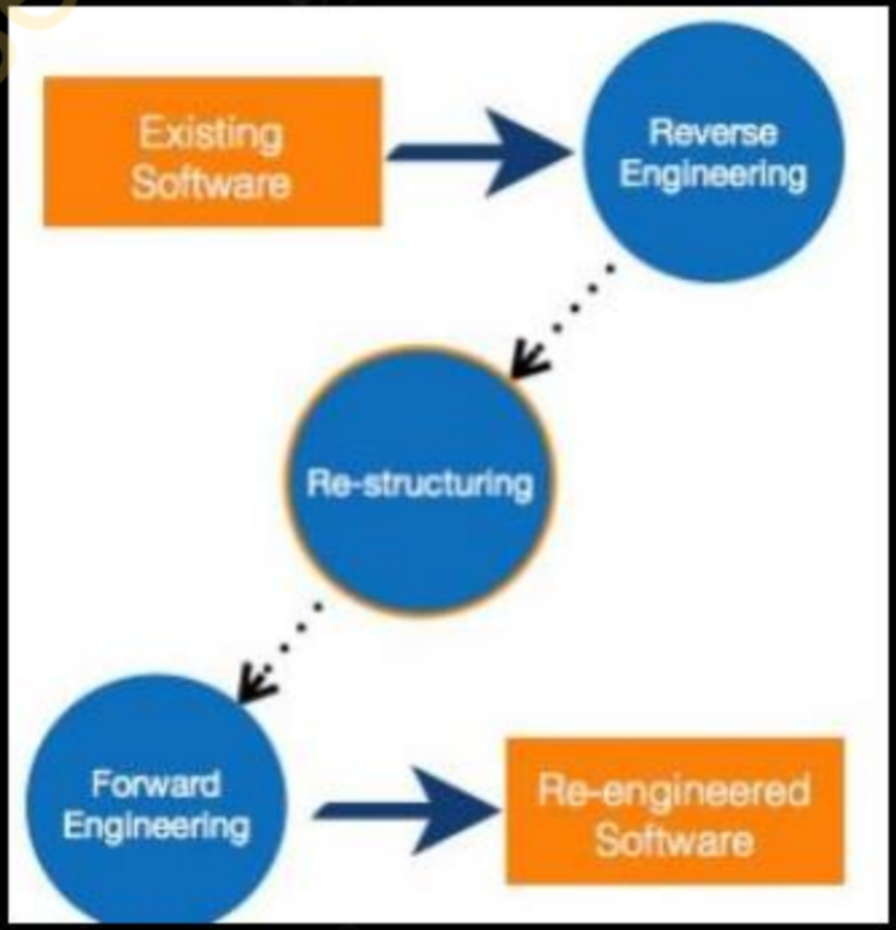
Delivery - After acceptance test, the system is deployed all over the organization either by small update package or fresh installation of the system. The final testing takes place at client end after the software is delivered. Training facility is provided if required, in addition to the hard copy of user manual.

Maintenance management - Configuration management is an essential part of system maintenance. It is aided with version control tools to control versions, semi-version or patch management.

Software Re-engineering

When we need to update the software to keep it to the current market , without impacting its functionality, it is called software re-engineering. It is a thorough process where the design of software is changed and programs are re-written.

For example, initially Unix was developed in assembly language. When language C came into existence, Unix was re-engineered in C, because working in assembly language was difficult. Other than this, sometimes programmers notice that few parts of software need more maintenance than others and they also need re-engineering.



Re-Engineering Process

- Decide what to re-engineer. Is it whole software or a part of it?
- Perform Reverse Engineering, in order to obtain specifications of existing software.
- Restructure Program if required. For example, changing function oriented programs into object-oriented programs.
- Re-structure data as required.
- Apply Forward engineering concepts in order to get re-engineered software. There are few important terms used in Software re-engineering.

Reverse Engineering

It is a process to achieve system specification by thoroughly analyzing, understanding the existing system. This process can be seen as reverse SDLC model, i.e. we try to get higher abstraction level by analyzing lower abstraction levels.

An existing system is previously implemented design, about which we know nothing. Designers then do reverse engineering by looking at the code and try to get the design. With design in hand, they try to conclude the specifications. So, going in reverse from code to system specification.

Program Restructuring

It is a process to re-structure and re-construct the existing software. It is all about re-arranging the source code, either in same programming language or from one programming language to a different one. Restructuring can have either source code-restructuring and data-restructuring or both.

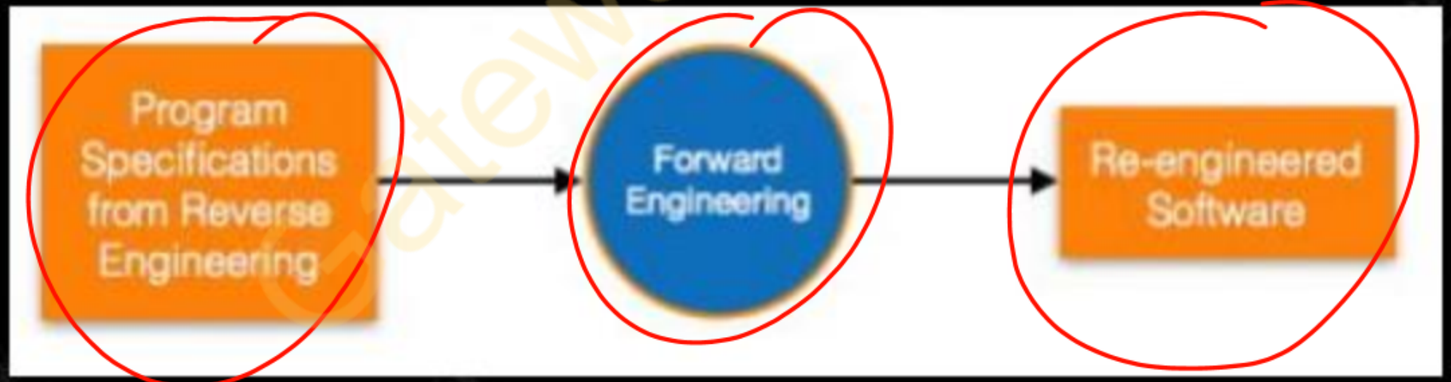
Re-structuring does not impact the functionality of the software but enhance reliability and maintainability. Program components, which cause errors very frequently can be changed, orupdated with re-structuring.

The dependability of software on obsolete hardware platform can be removed via re-structuring.

Forward Engineering

Forward engineering is a process of obtaining desired software from the specifications in hand which were brought down by means of reverse engineering. It assumes that there was some software engineering already done in the past.

Forward engineering is same as software engineering process with only one difference – it is carried out always after reverse engineering.



CASE Tools

CASE stands for Computer Aided Software Engineering. It means, development and maintenance of software projects with help of various automated software tools.

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.

Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

Components of CASE Tools

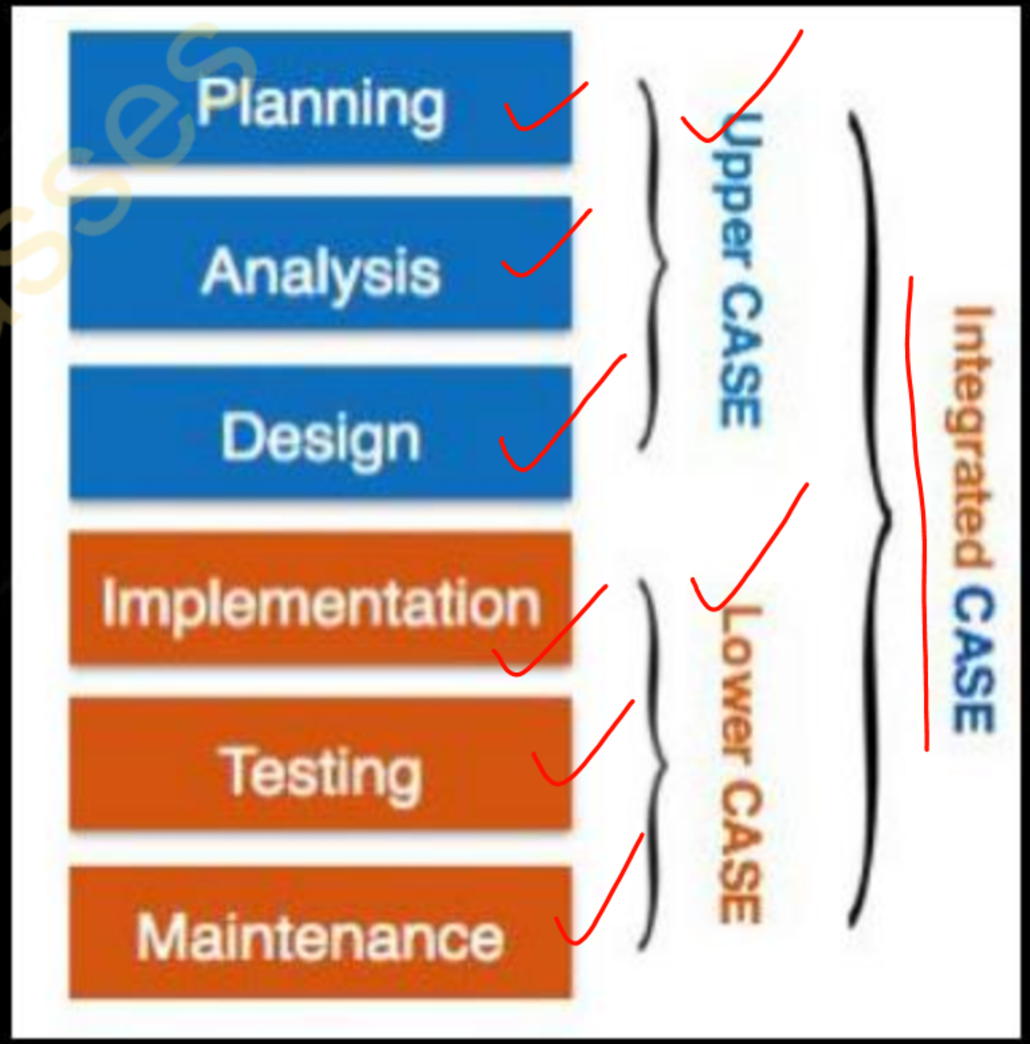
CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

Central Repository - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.

Upper Case Tools - Upper CASE tools are used in planning, analysis and design stages of SDLC.

Lower Case Tools - Lower CASE tools are used in implementation, testing and maintenance.

Integrated Case Tools - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.



Software Version Control

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

A version control system is a kind of software that helps the developer team to efficiently communicate and manage (track) all the changes that have been made to the source code along with the information like who made and what changes have been made.

Benefits of the version control system:

- Enhances the project development speed by providing efficient collaboration
- Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change
- Employees or contributors of the project can contribute from anywhere
- Helps in recovery in case of any disaster or contingent situation
- Informs us about Who, What, When, Why changes have been made

Project Estimation

Project size estimation is determining the scope and resources required for the project.

- It involves assessing the various aspects of the project to estimate the effort, time, cost, and resources needed to complete the project.
- Accurate project size estimation is important for effective and efficient project planning, management, and execution.

Importance of Project Size Estimation

Here are some of the reasons why project size estimation is critical in project management:

Financial Planning: Project size estimation helps in planning the financial aspects of the project, thus helping to avoid financial shortfalls.
Resource Planning: It ensures the necessary resources are identified and allocated accordingly.
Timeline Creation: It facilitates the development of realistic timelines and milestones for the project.
Identifying Risks: It helps to identify potential risks associated with overall project execution.
Detailed Planning: It helps to create a detailed plan for the project execution, ensuring all the aspects of the project are considered.
Planning Quality Assurance: It helps in planning quality assurance activities and ensuring that the project outcomes meet the required standards.

Project Risk Management

Risk management involves all activities pertaining to identification, analyzing and making provision for predictable and non-predictable risks in the project. Risk may include the following:

- Experienced staff leaving the project and new staff coming in.
- Change in organizational management.
- Requirement change or misinterpreting requirement.
- Under-estimation of required time and resources.
- Technological changes, environmental changes, business competition

Risk Management Process

There are following activities involved in risk management process:

- Identification - Make note of all possible risks, which may occur in the project.
- Categorize - Categorize known risks into high, medium and low risk intensity as per their possible impact on the project.
- Manage - Analyze the probability of occurrence of risks at various phases. Make plan to avoid or face risks. Attempt to minimize their side-effects.
- Monitor - Closely monitor the potential risks and their early symptoms. Also monitor the effective steps taken to mitigate or avoid them.

COCOMO Model

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e., the number of Lines of Code. The Cocomo Model is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models.

The key parameters that define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort and schedule:

1. Effort: Amount of labor that will be required to complete a task. It is measured in person months units.
2. Schedule: This simply means the amount of time required for the completion of the job,which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, and months.

Corrective Maintenance - This includes modifications and updations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.

Importance of the COCOMO Model

1. Cost Estimation: To help with resource planning and project budgeting, COCOMO offers a methodical approach to software development cost estimation.
2. Resource Management: By taking team experience, project size, and complexity into account, the model helps with efficient resource allocation.
3. Project Planning: COCOMO assists in developing practical project plans that include attainable objectives, due dates, and benchmarks.
4. Risk management: Early in the development process, COCOMO assists in identifying and mitigating potential hazards by including risk elements.
5. Support for Decisions: During project planning, the model provides a quantitative foundation for choices about scope, priorities, and resource allocation.
6. Benchmarking: To compare and assess various software development projects to industry standards, COCOMO offers a benchmark.
7. Resource Optimization: The model helps to maximize the use of resources, which raises productivity and lowers costs.

Gateway Classes

Empowering Learners, Trasforming Futures

Thank You

GET IT ON Google Play

Download on the App Store

Helpline No- 7819 0058 53